LISTING OF THE CLAIMS

At the time of the Action:

Pending Claims: 1-41

Withdrawn Claims: 39 and 40

Canceled Claims: None

After this Response:

Pending Claims: 1-14, 16-29, 31-38, 41, and 42

Amended Claims: 1-6, 8-14, 17-22, 24-29, 31-34, 36-38, and 41

Withdrawn: None

Canceled Claims: 15 and 30

New Claims: 42

1. (Currently Amended) In a host of a virtual machine environment having one or more methods in a shared managed library, a process ~~comprising allowing or disallowing a call~~ for managing calls from a first managed code caller to a first [[said]] method ~~based upon a configuration of the first said method with a hosting rule~~, the process managing calls based on a hosting rule selected from [[the]] a group ~~consisting~~ comprising of:

~~always allow~~ authorizing ~~any said call~~ calls from ~~any said~~ one or more of a plurality of managed code ~~caller~~ callers to the first [[said]] method;

~~never allow any said call~~ preventing calls from ~~any said~~ one or more of a plurality of managed code ~~caller~~ callers to the first [[said]] method due to the first method's inappropriateness for the ~~runtime~~ virtual machine environment; and conditionally ~~allow~~ authorizing ~~any said call~~ calls from ~~any said~~ one or more of a plurality of managed code ~~caller~~ callers to the first [[said]] method.

2. (Currently Amended) The process as defined in Claim 1, wherein the conditional ~~allowance~~ authorization is based upon:

the first method's required level of trust; and

a level of trust attributed to the first managed code caller.

3. (Currently Amended) The process as defined in Claim 2, wherein the ~~allowing or~~ authorizing and the ~~disallowing~~ preventing a call further comprises:

compiling code corresponding to the first managed code caller into native code; and

executing the native code corresponding to the first managed code caller ~~during which the call from~~ while the first managed code caller is making the call to the first [[said]] method native code ~~is made~~.

4. (Currently Amended) The ~~method~~ process as defined in Claim 3, further comprising throwing an exception [[ ]]during the ~~execution~~ executing ~~when the call from~~ and while the first managed code caller is making the call to the first [[said]] method native code ~~is made and~~ when:

the call [[that]] is ~~never allowed~~ prevented; or

the level of trust attributed to the first managed code caller is insufficient when compared to a security permission demand assigned to and required by the first [[said]] method.

5. (Currently Amended) The process as defined in Claim 1, wherein when the call from the first managed code caller is ~~allowed~~ authorized, access is provided by the first [[said]] method to a protected resource.

6. (Currently Amended) The process as defined in Claim 1, wherein any ~~said allowed~~ authorized call provides ~~any said~~ one or more of the plurality of managed code ~~caller~~ callers with access to one or more protected resources corresponding to the called [[said]] method.

7. (Original) The process as defined in Claim 1, wherein the level of trust attributed to the first managed code caller corresponds to an identity of a provider of the first managed code caller.

8. (Currently Amended) The process as defined in Claim 1, wherein the host compiles the first managed code caller into native code that is executed by a common language runtime via ~~the host's~~ an operating system of the host.

9. (Currently Amended) The process as defined in Claim 1, further comprising configuring each [[said]] method in the shared managed library with one [[said]] hosting

rule.

10. (Currently Amended) The process as defined in Claim 9, wherein each [[said]] method ~~receives the configuring~~ is configured prior to any [[said]] call to any [[said]] method from any [[said]] one of the plurality of managed code ~~caller~~ callers.

11. (Currently Amended) The process as defined in Claim 1, further comprising:

determining whether the host will use any [[said]] hosting rule in ~~allowing~~ authorizing a call from any [[said]] one of the plurality of managed code ~~caller~~ callers to any ~~said method~~ of the one or more methods; and

~~when the determination is affirmative,~~ configuring the one or more [[said]] methods in the shared managed library with one [[said]] hosting rule when the determination is affirmative, and not configuring the one or more methods in the shared managed library with one hosting rule when the determination is negative.

12. (Currently Amended) The process as defined in Claim 11, wherein:

each [[said]] method in the shared managed library provides access to one or more protected resources; and

the host has access to a host configuration data structure comprising:

resource checking data for making the determination;

configuration data referencing the one or more protected resources and specifying:

each [[said]] protected resource to which access will ~~always~~ be ~~allowed~~ authorized to any [[said]] one of the plurality of managed code ~~caller~~ callers;

each [[said]] protected resource to which access will [[never]] be ~~allowed~~ prevented to any [[said]] one of the plurality of managed code ~~caller~~ callers; and

each [[said]] protected resource to which access will be ~~allowed~~ authorized to any [[said]] one of the plurality of managed code ~~caller~~ callers having a recognized level of trust satisfying a security permission demand corresponding to the protected resource;

wherein the process further comprises:

accessing the host configuration data structure; and

using the resource checking data in the host configuration data structure to make the determination, wherein the configuring of the one or more [[said]] methods in the shared managed library with one [[said]] hosting rule comprises, for each [[said]] method:

matching each [[said]] protected resource to which the method provides access to the corresponding protected resource in the host configuration data structure; and

for each [[said]] match, assigning to the method the corresponding configuration data that is associated with the protected resource in the host configuration data structure.

13. (Currently Amended) A computer readable storage medium ~~including~~ having machine readable instructions stored thereon that, when executed by one or more processors, causes the one or more processors to ~~for implementing~~ implement the process as defined in claim 1.

14. (Currently Amended) A method, comprising:

intercepting, with a host operating in a managed environment, a call from a managed caller to a managed callee; and

deriving whether the call is permissible according to the host's prior configuration of a plurality of [[said]] managed callees, wherein:

each [[said]] managed callee provides access to a protected resource; and

the prior configuration specifies whether to:

~~always allow~~ authorize the call to be made;

~~never allow~~ prevent the call to be made; or

[[allow]] authorize the call to be made based upon the degree to which the host trusts the managed caller[[.]],

providing access to the protected resource to the managed caller when the call is permissible; and

preventing access to the protected resource to the managed caller when the call is not permissible.

15. (Canceled)

16. (Original) The method as defined in Claim 14, wherein the degree to which the host trusts the managed caller corresponds to an identity of a provider of the managed caller.

17. (Currently Amended) The method as defined in Claim 14, wherein the host compiles the managed caller into native code that is executed by a common language runtime via ~~the host's~~ an operating system of the host.

18. (Currently Amended) The method as defined in Claim 17, further comprising throwing an exception when~~, during the execution of the compiled native code corresponding to any said managed caller~~:

~~any said~~ the managed caller attempts to make a call that is ~~never allowed to be made is attempted~~ prevented; or

~~any said~~ the managed caller attempts to make a call ~~that is attempted~~ when the degree to which the host trusts the managed caller is insufficient.

19. (Currently Amended) The method as defined in Claim 14, further comprising, prior to the intercepting:

determining whether the host will ~~make the derivation~~ perform the deriving; [[and]]

performing the intercepting and the deriving if the determination is affirmative[[.]]; and preventing the intercepting and the deriving if the determination is negative.

20. (Currently Amended) The method as defined in Claim 19, wherein:

the host has access to a host configuration data structure comprising:

resource checking data for making the determination; and

configuration data sufficient for the host's prior configuration of the plurality of [[said]] managed callees;

the determining whether the host will make the derivation comprises accessing, with the host, the resource checking data in the host configuration data structure.

21. (Currently Amended) A computer readable storage medium ~~including~~ having machine readable instructions stored thereon that, when executed by one or more processors, causes the one or more processors to ~~for implementing~~ implement the method as defined in claim 14.

22. (Currently Amended) An apparatus, comprising:

virtual machine means, in a managed code portion including a plurality of ~~method~~ methods in a shared managed library, for operating a plurality of managed code callers in the managed code portion;

execution engine means, in a native code portion, for the virtual machine means;

means, in a native code portion, for providing a runtime engine in an operating system; and means for ~~allowing~~ authorizing or ~~disallowing~~ preventing a call from a first [[said]] one of the plurality of managed code ~~caller~~ callers to a first [[said]] one of the plurality of ~~method~~ methods based upon a configuration of the first [[said]] method with a hosting rule selected from [[the]] a group ~~consisting~~ comprising of:

~~always allow~~ authorizing ~~any said call~~ calls from any [[said]] one of the plurality of managed code ~~caller~~ callers to the first [[said]] method;

~~never allow any said call~~ preventing calls from any [[said]] one of the plurality of managed code ~~caller~~ callers to the first [[said]] method due to the first method's inappropriateness for the runtime environment; and

conditionally [[allow]] authorizing ~~any said call~~ calls from any [[said]] one of the plurality of managed code ~~caller~~ callers to the first [[said]] method based upon:

~~the configurations an assignment of the~~ a method's required level of trust; and

a level of trust attributed to the managed code caller.

23. (Original) The apparatus as defined in Claim 22, wherein the level of trust attributed to the managed code caller is based upon an identification (ID) of the provider of the managed code caller.

24. (Currently Amended) The apparatus as defined in Claim 22, further comprising:

means for compiling each [[said]] one of the plurality of managed code ~~caller~~

callers from an intermediate language code and metadata into native code;

means for loading the native code with a Common Language Runtime (CLR) loader in the native code portion to load the compiled native code; and

means for executing the compiled native code in the native code portion ~~such that each said~~ causing the managed code caller [[can]] to call one [[said]] method.

25. (Currently Amended) The apparatus as defined in Claim 22, further comprising means for throwing an exception when [[a]] one of the plurality of managed code callers attempts to make a ~~disallowed~~ prevented call ~~is attempted~~ during the execution of the compiled native code corresponding to any [[said]] one of the plurality of managed code ~~caller~~ callers.

26. (Currently Amended) The apparatus as defined in Claim 22, wherein the managed code portion further comprises one or more files associated with user code that, when compiled into an intermediate language code and metadata generated by a language compiler, are represented by one or more of [[said]] the plurality of managed code callers.

27. (Currently Amended) The apparatus as defined in Claim 22, wherein the execution engine means in the native code portion further comprises a compiler to compile each [[said]] one of the plurality of managed code ~~caller~~ callers into native code for execution by the native code portion.

28. (Currently Amended) The apparatus as defined in Claim 22, wherein the execution engine means in the native code portion further comprises:

a Just In Time (JIT) compiler to compile each [[said]] one of the plurality of managed code ~~caller~~ callers into native code; and

a CLR loader to load the compiled native code for execution by the native code portion.

29. (Currently Amended) A computing device, comprising:

a managed code portion including:

one [[of]] or more methods in a shared managed library;

one or more assemblies placed in respective application domains for execution; and

a virtual machine;

a native code portion including:

an execution engine for the virtual machine; and

an operating system under the execution engine;

logic configured to:

intercept a call from one [[said]] assembly to one [[said]] method; [[and]]

~~deriving~~ derive whether the call is permissible according to a prior configuration of the one of more methods, wherein:

each [[said]] method provides access to a protected resource; and

the prior configuration specifies whether to:

~~always allow~~ authorize the call to be made;

~~never allow~~ prevent the call to be made;

[[allow]] authorize the call to be made based upon the degree to which the one [[said]] assembly is trusted by the computing ~~devices~~ device[[.]];

provide to the one assembly access to the corresponding protected resource when the call is permissible; and

prevent access to the one assembly to the corresponding protected resource when the call is not permissible.

30. (Canceled)

31. (Currently Amended) The computing device as defined in Claim 29, wherein the degree to which the host trusts the managed caller corresponds to an identity of a provider of the one [[said]] assembly.

32. (Currently Amended) The computing device as defined in Claim 29, wherein the computing device compiles the one [[said]] assembly into native code that is executed by a common language runtime via the operating system.

33. (Currently Amended) The computing device as defined in Claim 32, further comprising throwing an exception when~~, during the execution of the compiled native code corresponding to the one said assembly~~:

the prior configuration specifies to attempt to make the call that is ~~never allowed to be made is attempted~~prevented; or

the prior configuration specifies to attempt to make the call ~~that is attempted~~ when the degree to which the computing device trusts the one [[said]] assembly is insufficient.

34. (Currently Amended) The computing device as defined in Claim 29, further comprising, prior to the intercepting:

determining whether the computing device will make the derivation; [[and]]

performing the intercepting and the deriving if the determination is affirmative[[.]]; and

not performing the intercepting and the deriving if the determination is negative.

35. (Original) The computing device as defined in Claim 34, wherein:

the computing device has access to a host configuration data structure comprising:

resource checking data for making the determination; and

configuration data sufficient for the computing device's prior configuration of the one of more methods;

the determining whether the computing device will make the derivation comprises accessing, with the computing device, the resource checking data in the host configuration data structure.

36. (Currently Amended) The computing device as defined in Claim 29, wherein the logic is further ~~configured~~ to receive intermediate language code and metadata generated by a language compiler to form the one or more assemblies for placement within respective application domains for execution.

37. (Currently Amended) The computing device as defined in Claim 36, wherein the intermediate language code and the metadata generated by the language compiler are generated from one or more files each having a file type and each being associated with user code.

38. (Currently Amended) The computing device as defined in Claim 29, wherein the execution engine further comprises:

a JIT complier to compile [[said]] the assemblies into native code; and

a CLR loader to load the compiled native code for execution in the native code portion.

39. (Withdrawn) A host comprising logic for a runtime environment using a host configuration data structure containing host configuration information to disallow a call to a managed code callee from an untrusted managed code caller or to disallow a call to a managed code callee that is deemed inappropriate for the runtime environment.

40. (Withdrawn) The host as defined in Claim 39, wherein:

the host compiles the managed code caller into native code that is executed by a common language runtime via the host's operating system; and

a host protection exception is thrown during the execution of the native code when the call to the managed code callee:

is made by the untrusted managed code caller; or

is deemed inappropriate for the runtime environment.

41. (Currently Amended) A host operating in a managed environment [[and]], comprising:

logic for configuring each of a plurality of managed callees, each providing access to a protected resource, with a configuration that:

~~always allow~~ authorizes a call to be made to each of the plurality of managed ~~callee~~ callees for access to the corresponding protected resource;

~~never allow~~ prevents a call to be made to each of the plurality of managed ~~callee~~ callees for access to the corresponding protected resource; or

~~allow~~ authorizes a call to be made to each of the plurality of managed ~~callee~~ callees for access to the corresponding protected resource based upon trust of the host for one of a plurality of managed ~~caller~~ callers;

logic for intercepting a call from a particular [[said]] one of the plurality of managed ~~caller~~ callers to a particular [[said]] one of the plurality of managed ~~callee~~ callees; [[and]]

logic, after intercepting the call, for determining whether the call is permissible according to the configuration of the particular one of the plurality of managed ~~callee~~ callees[[.]]; and

logic, after determining whether the call is permissible, for either providing access to the particular one of the plurality of managed callers to the protected resource when the call is permissible or preventing access to the particular one of the plurality of managed callers to the protected resource when the call is not permissible.

42. (New) The process as defined in Claim 1, wherein the managing calls comprises either authorizing or preventing a call from a first managed code caller to a first method based at least in part on the first method.